**INFORMATION SYSTEMS SERVICES**

# White Rose Grid Node 1 (MAXIMA) User Guide

***This is a Getting Started document for new users of White Rose Grid Node 1, known as the MAXIMA. It contains information for users of the Sun Fire cluster. Please read it carefully before attempting to login and use the system.***

**AUTHOR:** Dr Joanna Schmidt, ISS, University of Leeds

**DATE:** Updated by Dr. A. N. Real: October 2003

**EDITION:** 2.0; © Copyright 2003,J.G.Schmidt

**FREE**

## Contents


1 Introduction ..... 3
1.1 About the WRG Grid Node 1 ..... 3
1.2 Becoming a user ..... 3
1.3 Connecting, logging into and logging out of the system ..... 3
2 Resource allocation ..... 4
2.1 Disk space ..... 4
2.2 CPU time ..... 4
2.3 Other resources ..... 5
3 Software development environments and tools ..... 5
3.1 Compilers ..... 5
3.1.1 An example of compilation and execution of a serial Fortran program ..... 7
3.1.2 An example of compilation and execution of a serial C program ..... 7
3.1.3 An example of compilation and execution of a java program ..... 7
3.2 64-bit application development environment ..... 7
3.3 Libraries and other tools ..... 7
3.4 Sun Cluster Runtime Environment ..... 8
3.5 MPI - Message Passing Interface ..... 9
3.5.1 An example of compilation and execution of a parallel MPI Fortran program ..... 9
3.6 OpenMP ..... 9
3.6.1 An example of compilation and execution of a parallel OpenMP Fortran program ..... 10
3.7 The Shell ..... 10
3.8 Editors ..... 11
3.9 Debuggers ..... 11
3.10 Profiling tools ..... 11
3.11 Printing ..... 12
3.12 Accessing your Origin2000 files ..... 12
4 Using the Sun Fire cluster ..... 13
4.1 Interactive access ..... 13
4.2 Batch jobs ..... 13
4.2.1 About SGEEE ..... 13
4.2.2 SGEEE queues ..... 13
4.2.3 Policies for job prioritisation ..... 14
4.2.4 Submitting batch jobs to SGEEE ..... 14
4.2.5 Submitting jobs using qsub ..... 14
4.2.6 Job output ..... 16
4.2.7 An example of an MPI job submission to SGEEE ..... 16
4.2.8 An example of OpenMP job submission to SGEEE ..... 16
4.2.9 An example of array job submission to SGEEE ..... 17
4.3 Interactive SGEEE jobs ..... 17
4.4 Querying queues ..... 18
4.5 Job deletion ..... 19
4.6 The GUI `qmon` command ..... 19
4.7 Usage accounting statistics ..... 19
5 On-line Information ..... 20
6 Help and user support ..... 20
7 Emailing list ..... 20
8 Code of Conduct ..... 20
9 Hints ..... 20
10 Links* ..... 21
Appendix A ..... 22

# 1 Introduction

This document contains information for new users of the White Rose Grid Node 1 service at the University of Leeds. The document explains how to apply for a username on the White Rose Grid Node 1 facility, known as the **`maxima`**, how to get access to the system, and gives the necessary information required to start using the service.

The **`maxima`** is part of the White Rose Grid facilities which are managed jointly with our two partners from the White Rose universities i.e. Sheffield and York. The White Rose Grid (WRG) Consortium, which operates under the auspices of the White Rose University Consortium, comprises those researchers from the three White Rose universities whose computational research requires access to leading-edge technology computers.

The White Rose Grid equipment has been acquired from, delivered and installed by Esteem Systems plc together with Sun Microsystems, and Streamline Computing Ltd.

These new systems, which are located at the University of Leeds, are operated and supported by the Information Systems Services staff on behalf of the White Rose Grid Consortium. Information on using the maxima facility is given below; for further assistance please contact ISS Helpdesk via email to helpdesk@leeds.ac.uk or telephone 0113 343 3333.

## *1.1 About the WRG Grid Node 1*

The WRG Node 1 computational facility is a cluster of Sun Fire servers manufactured by Sun Microsystems, Inc. This is a constellation of symmetric multiprocessor systems (SMPs) with shared-memory. It comprises a Sun Fire 6800 with 20 UltraSPARC III Cu 900 MHz processors, 44 GB of physical memory, and 100 GB storage. In addition, this WRG node includes five Sun Fire V880 servers, each with 8 UltraSPARC III Cu 900 MHz processors, 24 GB RAM, and twelve 36 GB FC-AL disks. Gigabit Ethernet serves as the cluster's interconnect. WRG Node 1 and Node 2 are attached to a shared filestore that provides 2 TB of usable disk space.

The computers offer the Solaris 8 operating system environment. Sun HPC ClusterTools software and Sun Forte Developer products are installed on all systems. The batch processing capabilities are provided by the Sun Grid Engine, Enterprise Edition product.

## *1.2 Becoming a user*

To register, users are required to complete the ISS Application Form for a Computer Username. The completed form must be signed by the WRG Node 1 representative and handed in at the ISS Helpdesk.

Note that once you have been registered, your allocated username and password will be sent to your WRG Node 1 representative for you to collect.

## *1.3 Connecting, logging into and logging out of the system*

The system is connected to the Leeds University campus network via a 100 Mbit/s Ethernet switch and can be accessed from any networked computer. You can use a variety of terminal types, e.g. workstations, PCs, that support TCP/IP, to connect to the system. The hostname is `maxima.leeds.ac.uk` and the `ip` address is `129.11.33.225`

You may use the `rlogin` program, available on many UNIX systems, to access this Sun Fire cluster. If you have access to an X-windows capable display then you may prefer, at the time of logging in, to establish an X session. In this case you may first need to allow access to your display using the `xhost` command in your terminal window on the workstation, by issuing the following command on your terminal before logging in to the system:

```
% xhost  +maxima.leeds.ac.uk
```

Then after logging in, set the environment variable `DISPLAY` correctly, i.e. type the following command:

```
% setenv DISPLAY  workstation_name.leeds.ac.uk:0.0
```

Where `workstation_name` is the mnemonic name (e.g. **`sgi044`**) or the IP address of your terminal. The `DISPLAY` environment variable can be set permanently in your `.login` file.

Alternatively, if you prefer to use the secure shell then simply issue the following command:

```
% ssh -X username@maxima.leeds.ac.uk
```

This method of access will automatically allow you to execute the various X-based software products, for example `prism`, without the need to set up the display variables manually on the local or remote machines. Furthermore, secure shell gives the functionality of using secure file copy, `scp`, to transfer files to and from the maximas.

Once your connection has been established you will be prompted for a username and password, which you must collect from your WRG Node 1 (`maxima`) representative. When logged on, you should change your initial password with the command:

```
% passwd
```

To leave the maxima system, type:

```
% logout
```

# 2 Resource allocation

The White Rose Grid project is a collaborative venture between the three White Rose universities. A certain proportion of resources is shared between the three institutions. WRG Node 1 allocates 75% of its resources equally to the seven shareholding groups from the University of Leeds; and the remaining 25% of total resources are to be allocated to WRG collaborative projects.

## *2.1 Disk space*

Your main working directory on Unix is known as your home directory, which can also be referred to as `$HOME`.

Disk storage for user home directories and software applications is provided by Sun StorEdge T3 Fibre Channel disk technology. At present we have one rack with 4 StorEdge T3 disk arrays. Both WRG Nodes 1 and 2 are attached to a shared filestore that provides 2 TB of usable disk space.

The storage resource is managed by the SAMFS hierarchical storage management filesystem. This manages files in two storage levels – a cache on disks and an archive on removable media such as tape. Within this filesystem, copies of files on disk are taken for backup and disk space is freed up by automatically moving old files to tape. Consequently, the restoration of deleted files is more convenient than retrieving backup from tape storage.

## *2.2 CPU time*

All CPU usage is recorded and is shown in the usage accounting reports that are displayed on a per-month, per department basis at http://www.leeds.ac.uk/iss/wrgrid/Usage.

## 2.3 Other resources

Memory use and disk i/o transfers are also recorded and may be reported in the future.

# 3 Software development environments and tools

The operating system on the `maxima` is a version of Unix called Solaris, the Sun implementation of Unix V Release 4 (SVR4). It provides full facilities for developments, compilation and execution of programs. A list of some useful Unix commands is available in Appendix A.

## 3.1 Compilers

The following compilers are available on the Sun Fire cluster:

| **Compiler** | **Description** |
|---|---|
| Fortran95 (90) | Forte Developer 7 (Sun WorkShop) Fortran 95 (90) compiler |
| Fortran 77 | Forte Developer 7 (Sun WorkShop) Fortran 90/95 compiler invoked with Fortran 77 backward compatibility. |
| C | Forte Developer 7 (Sun WorkShop) C compiler |
| C++ | Forte Developer 7 (Sun WorkShop) C++ compiler |
| Java | Java compiler |

Table 1

The actual compilers (and the loader) are simply called by issuing the `f90, f95, f77, cc, CC` or `javac` commands for Fortran 90, Fortran 95, Fortran77, C, C++, and java respectively. The Fortran, C, and C++ compilers process OpenMP shared-memory multiprocessing directives.

Fortran programmers should note that the suffix extension appearing on your program determines how the compiler processes the file.

The compilers' features are selectable by optional flags specified on the command line. If conflicting options are defined on the same compilation line, the right-most option specified has precedence. Perhaps the most commonly used options for Fortran code compilation are:

| **Fortran compiler flags** | **Action** |
|---|---|
| `-fast` | Optimise code using a set of predetermined options. Specify this flag before the following switches `-xchip, -xarch, -xcache` on the command line. |
| `-c` | Compile only; suppress linking. |
| `-o` *file_name* | The name of the executable file instead of `a.out` |
| `-xarch=v8plusb`<br>`-xarch=v9b` | Use the `-xarch=v8plusb` option for 32-bit addressing, and `-xarch=v9b` for 64-bit addressing. |
| `-xchip=ultra3cu` | Create the executable for UltraSPARCIII Cu processors. |
| `-xcache=64/32/4:8192/512/2` | Specify the cache configuration on the UltraSPARCIII in order for the cache optimisations to be carried out. |

| | |
|---|---|
| `-g` | Use this option to produce the code for debugging and/or source code commentary for profiling i.e. the driver will produce additional symbol table information. |
| `-pg` | Prepare for profiling by statement or procedure. |
| `-XlistL` | Generate source listing and errors. |
| `-Xlist` | Used for debugging i.e. global program checking across routines for consistency of arguments, commons etc.; includes source listing. |
| `-O level` | Specify optimisation level. Note that the highest optimisation level is 5 (`-O5`). |
| `-u` | Check for any undeclared variables. |
| `-C` | Check at runtime for out-of-bounds references in each array subscript. |
| `-help` | Summaries of the command line options are supplied. |
| `-xhelp=readme` | View Forte Developer 7 README file. |
| `-autopar`<br>`-xautopar` | Enable automatic loop parallelisation. |
| `-explicitpar` | Enable parallelization of loops or regions explicitly marked with parallel directives. To improve performance, also specify the `-stackvar` flag when using any of the parallelisation options. |
| `-openmp`<br>`-xopenmp` | Accept OpenMP API directives and set appropriate environment. |
| `-stackvar` | Allocate all local variables on the stack. |
| `-parallel`<br>`-xparallel` | Parallelise loops with `-autopar, -explicitpar, -depend` |
| `-mp=sun`<br>`-mp=cray`<br>`-mp=openmp` | Option to select the style of parallelisation directives enabled: Sun, Cray, or OpenMP. |

Table 2

Please note that when compiling and linking in separate stages, identical compiler options should be used in each case. Furthermore, when compiling an executable from multiple source files, some compiler options must be consistent for all source files at both the compile and link stages.

All compilers and their respected options are documented in the **`man pages`** which are invoked by typing for example:

```
% man f95
```

Other sections, for example denoted by ieee_flags(3M), are accessed using the `-s` flag on the `man` command, i.e.:

```
% man -s 3M ieee_flags
```

### 3.1.1 An example of compilation and execution of a serial Fortran program

Assuming that the program source code is contained in the `mycode.f` file to compile this code using the Fortran 95 compiler type:

```
% f95 –fast mycode.f
```

In this case the executable code will be output into the file `a.out`. To run this code interactively, type after the prompt:

```
% a.out
```

### 3.1.2 An example of compilation and execution of a serial C program

Assuming that the program source code is contained in the `myprogram.c` file, to compile this code using the C compiler type:

```
% cc –o myprogram myprogram.c
```

In this case the executable code will be output into the file `myprogram.` To run this code interactively, type after the prompt:

```
% myprogram
```

For optimisation you may wish to use the following switches: `-fast`, `-xarch=[v8plusb|v9b]` and `–xcache=64/32/4:8192/512/2.` Also you may wish to use `–xdepend, -xprefetch=yes, -xvector=yes` and `-xsfpconst.` See the **`man`** pages or type **`cc -help`** for details.

### 3.1.3 An example of compilation and execution of a java program

The java code contained in a file `myprogram.java` may be compiled as follows:

```
% javac -O myprogram.java
```

and run:

```
% java myprogram
```

## *3.2 64-bit application development environment*

Sun Forte Developer products offer support for the development of both: 32-bit and 64-bit applications. Note that the 64-bit technology will allow a user to use the 64-bit address space which increases the capacity of problems you can consider, offers 64-bit integer arithmetic (with increased speed of calculations for mathematical operations), and supports the use of larger files (greater than 4 GB).

To build a 64-bit executable you must specify the `–xarch=v9b` option when compiling and linking your code (`-xarch=v8plusb` should be used for 32-bit addressing).

## *3.3 Libraries and other tools*

| **Software product** | **Description** |
|---|---|
| MPI (part of HPC Cluster Tools) | Library for developing message-passing programs. |
| OpenMP API | API for developing shared-memory programs. |

| | |
|---|---|
| Sun Forte Developer (Sun WorkShop) | Offers an integrated programming environment for the development of shared-memory applications. |
| Sun WorkShop Visual 6 | Tools to create C++ and Java graphical user interfaces. |
| Sun ONE Studio 4 | Integrated development environment for Java applications. |
| Sun Performance Library | The optimised library of subroutines and functions used for linear programming and FFT; based on the standard libraries LAPACK, BLAS1, BLAS2, BLAS3, FFTPACK, VFFTFPACK and LINPACK.<br><br>To link with the Sun performance library you must compile using the flag `-dalign` (which is included in the `-fast` macro) and link using the option `-xlic_lib=sunperf`. |
| Sun Scalable Scientific Subroutine Library (Sun S3L) – (part of HPC ClusterTools) | This library provides a set of parallel functions and tools for MPI programs written in Fortran77/90, C and C++.<br>See **`man s3l`** for details of routines and use. |
| NAG Fortran Library | This is the Numerical Algorithms Group's Fortran 77 library. Note that, except for the two routines X04ACF and X04ADF, the libraries in this implementation are compatible with Sun Fortran 90/95, provided that the f90/95 compiler is called with the flag -lF77 (and not -lf77compat ).<br><br>Please note: only the 32-bit library is available at present.<br><br>To compile and link with the library add the `-dalign` and `-lnag` compiler flags. If you are compiling with `-fast`, the `-dalign` flag may be omitted but better performance may be obtained by linking with the `-lnag-spl` option, together with the Sun Performance library `-xlic_lib=sunperf`. |
| Prism (part of HPC ClusterTools) | Provides a graphical programming environment to develop, execute, debug and visualise data in message-passing programs written in Fortran 77, Fortran 90, C and C++. Prism must be invoked from an X-windows display on the system. |
| Sun CRE | The Sun Cluster Runtime Environment (CRE) manages the resources of a cluster nodes. It manages launching and execution of both serial and parallel jobs on the cluster nodes. |

Table 3

Importantly, please note that the Sun Forte Developer software, which runs in X-windows, is invoked by typing:

```
% workshop
```

## 3.4 Sun Cluster Runtime Environment

The Cluster Runtime Environment (CRE) is a component of Sun HPC ClusterTools software. It manages the resources of a cluster to execute message-passing programs.

The CRE environment offers the following important components:

| Command | Action |
|---|---|
| mprun | run MPI programs |
| mpps | display status information about jobs executing |
| mpkill | kill programs |

Table 4

To run the program as multiple processes with MPI calls use the following syntax:

```
% mprun -np number_of_processes program_name
```

To display status information about your jobs type:

```
% mpps
```

To kill a running program type:

```
% mpkill job_id
```

To display the help/usage text please invoke these three commands (`mprun`,`mpps`,`mpkill`) with the flag `-h`

## 3.5 MPI - Message Passing Interface

MPI (Message Passing Interface) is a specification for the user interface to a message-passing library used for writing parallel programs. It was designed by a broad group of parallel computer vendors, library writers, and application developers to serve as a standard. MPI is implemented as a library of routines which can be used for the development of portable Fortran, C and C++ programs to be run across a wide variety of parallel machines, including massively parallel supercomputers, shared-memory multiprocessors, and networks of workstations.

Sun MPI is a library of message-passing routines compliant with the MPI 1.1 standard and partially with MPI 2. The `mpf77, mpf90, mpf95, mpcc,` and `mpCC` utilities may be used to compile Fortran77, Fortran90, Fortran95, C and C++ programs respectively.

### 3.5.1 An example of compilation and execution of a parallel MPI Fortran program

For example, assuming that the source code is contained in the file `mycode.f,` to compile this program using the Fortran 77 compiler and produce the executable file, type:

```
% mpf77 -fast mycode.f  -lmpi
```

In this case the executable code will be created in the file `a.out.` To run this code interactively on 2 processors under the Sun Cluster Runtime Environment (CRE) type after the prompt:

```
% mprun -np 2 a.out
```

## 3.6 OpenMP

OpenMP offers the API (application programming interface) standard for parallel programming on multi-platform shared-memory computers. It supports the shared-memory parallel programming model and thus provides a simple yet powerful model to the programmer for expressing and managing parallelism in an application. It allows a user to create and manage parallel programs while ensuring portability across shared-memory parallel systems.

OpenMP is available to Fortran 90 (Fortran95) and C/C++ software developers in the Sun Forte Developer 7 (WorkShop) environment.

### 3.6.1 An example of compilation and execution of a parallel OpenMP Fortran program

For example, assuming that the program's source code is contained in the file `mycode.f90` to compile this code using the Fortran90 compiler type:

```
% f90 -fast -openmp -stackvar mycode.f90
```

The file `mycode.f90` contains the following source code:

```
       program hello
       integer :: OMP_GET_THREAD_NUM, tid
!$OMP PARALLEL
       tid = OMP_GET_THREAD_NUM()
       print *, 'my thread id is', tid
!$OMP END PARALLEL
       end
```

In this case the executable code will be created in the file `a.out.` To run this code interactively on 2 processors type after the prompt:

```
% setenv OMP_NUM_THREADS 2
% a.out
```

The output of this executable is as follows:

```
my thread id is 0
my thread id is 1
```

## 3.7 The Shell

The C shell (csh) is the default shell on the cluster. For this shell the basic setup file is called `.cshrc.` Should you wish to change the basic behavior of this shell change the `.cshrc` file. A csh executes the `.login` file and then the `.cshrc` file when you log in, and the `.logout` file when you logout out. These files are located in your home directory and to see them type:

```
% ls -la
```

In scripts the C shell is called by the following sequence in the first line:

```
#!/bin/csh
```

To setup the environment variables which control the shell's behavior type:

```
% setenv  variable_name value
```

where the `variable_name` is the name of the environment variable, and value is the value it is to be set to.

The shell is documented in the `man` pages; type **`man csh`** to get more details of this shell.

## 3.8 Editors

The following Unix editors are available on the system:

```
vi
nedit
emacs
```

A fact card for the **vi** editor is available at the ISS Documentation pages at: http://www.leeds.ac.uk/iss/documentation . To invoke **vi** in order to create or edit a file type:

```
% vi filename
```

If the specified file does not already exist a new file will be created. If the file already exists, it will be copied into the edit buffer. To terminate the edit and save the information, press escape then type:

```
:wq
```

NEdit is a GUI style editor for plain text files. It requires an X-Windows system based workstation or X-terminal. To use the `nedit` editor type:

```
% nedit filename
```

For further information type:

```
% man nedit
```

## 3.9 Debuggers

The following debuggers are available on the system:

| | |
|---|---|
| `dbx` | Standard UNIX debugger (see **`man dbx`**) with command line interface. |
| `prism` | A graphical debugger that works in the Common Desktop Environment or OpenWindows and X windows. |
| `workshop` | The Sun Forte (WorkShop) integrated programming environment allows you to edit, build, debug, analyze, and browse a program without having to explicitly start individual tools from the command line. |

## 3.10 Profiling tools

The following standard Unix program performance evaluation tools are available on the `maxima` system:

| | |
|---|---|
| `prof` | Profiling tool. |
| `gprof` | Graphical version of `prof` that works in the Common Desktop Environment or OpenWindows. |

The following profiling tools are also available:

| | |
|---|---|
| `prism` | A graphical MPI profiling tool. |
| `workshop` | The Sun Forte (WorkShop) integrated programming environment allows you to debug and profile a program without having to explicitly start individual tools from the command line. |

| | |
|---|---|
| `analyzer` | GUI that can be used to visualize profiling statistics produced with the collect utility. |
| `collect` | Tool to produce profile statistics when running an executable. These statistics can be viewed using the `analyzer` tool, or with the `er_print` command line utility.<br><br>Note that when invoking MPI code, the `collect` command should follow the `mprun` command and arguments, before the executable name. |
| `er_src` | Tool to print out source code compiler commentary on an object file compiled with the `-g` flag. |
| `er_print` | Command line tool to print out profiling statistics, without using the `analyzer` GUI. |

## 3.11 Printing

You may print to any of the ISS printers by typing:

```
% lpr -Pprinter_name file_name
```

The **`lpq`** command shows the status of a printer, for example the list of jobs in the queue.

## 3.12 Accessing your Origin2000 files

Users may access their files, which were created on the Origin2000 system, by first typing the following command:

```
% cd $HPC
```

This command will work for those users who have the same username on both systems.

Those users who have the same username on both system may use the following two commands on the `maxima` to transfer files from the Origin2000 filestore to the `maxima` filestore:

```
% cd  $HPC
% tar cvBf - * |(cd ~; tar xpBf -)
```

These two commands will move all your files with the exception of **`.*`** files. This means that your **`.profile`**, **`.login`** and **`.cshrc`** files will not be overwritten.

Please note: that if you wish to use these commands to transfer your files from the Origin 2000 to the `maxima` system then this `tar` command **must be the very first thing to be issued** when you login to the system for the first time. Otherwise you may overwrite your files. If you issue these two commands again you may overwrite files in your home directory on the maxima system.

# 4 Using the Sun Fire cluster

The cluster is configured with a front-end server (one of the V880 systems) and back-end computers which comprise the Sun Fire 6800 and the four remaining V880 servers.

Users are allowed only to login into the front-end server where they can develop their programs and from where they can submit their jobs to the back-end(s) computers. The front-end system may be used interactively for editing, compilation and debugging of users' programs. This system is also a submission host for executing SGEEE jobs on the other systems.

Direct interactive access to the back-end systems is not allowed. The Sun Fire 6800 and the four remaining V880 servers are configured as separate systems with separate sets of queues.

## *4.1 Interactive access*

To ensure the effective use of WRG Node 1 resources, the batch processing systems, Sun Grid Engine, Enterprise Edition (SGEEE), has been installed on these servers. The job manager allows system resources to be allocated in a controlled manner to batch requests, and should be used to execute all production runs as well as to execute some of the development codes.

At present, users are advised to edit their files, compile their programs and run interactively programs executing in a short time (not more than in 15 minutes). Interactive jobs exceeding the specified limit (15 mins or 4 processors) may be terminated as they may affect the performance of the system.

Users may submit interactive jobs to SGEEE which runs them in the high priority interactive queues created by the administrator.

## *4.2 Batch jobs*

Batch processing is an important service that is controlled by the Sun Grid Engine, Enterprise Edition product (SGEEE).

### 4.2.1 About SGEEE

The Sun Grid Engine, Enterprise Edition product is a resource management tool which might be used to enable grid computing. This is a complex and powerful package. Grid Engine is an advanced batch processor that schedules jobs, submitted by users, to appropriate systems available under its configuration according to the resource management policies accepted by the organisation. It manages global resource allocation (CPU time, memory, disk space) across all systems under its control. SGEEE controls the delivery of computational resources by enforcing policies set by the administrator.

### 4.2.2 SGEEE queues

Batch and interactive jobs may be submitted to SGEEE. All jobs submitted to SGEEE, with the exception of interactive ones, will be held in a spool area waiting for the scheduling interval when a scheduler dispatches jobs for processing on the basis of their ticket allocations. Tickets are used to enforce scheduling policies. The more tickets the job is assigned the more important the job is and it is dispatched preferentially. Jobs accumulate tickets from all policies. If no tickets are assigned to the policy then the policy is not used. At each scheduling period the number of tickets owned by each job, including the executing jobs, is re-evaluated. Jobs currently executing are also evaluated at each scheduling period and their allocation of tickets may be amended. Tickets assigned by the administrator enable the scheduler to determine which jobs should be run next.

Users submit jobs to a queuing system and the scheduler allocates them to the relevant queue. The current queue configuration, which includes details of job limits, is available from http://www.leeds.ac.uk/iss/wrgrid/Documentation/Node1queues.html.

### 4.2.3 Policies for job prioritisation

There are four policies that can be applied by SGEEE to schedule users' jobs. These are as follows:

- Share-based (also called share tree) - when this policy is implemented users are assigned the level of service according to the share they own and the past usage of resources by all users and their intended use of systems. It allows for share entitlements to be implemented in a hierarchical fashion.

- Functional - when this policy is implemented users are assigned the level of service according to a share they own and the current presence of other jobs. This policy is similar to the shared-based policy but does not consider the past usage of the system. It allows for share entitlements to be implemented in a hierarchical fashion.

- Deadline - this policy assigns high priority to certain jobs that must be finished before the deadline.

- Override - this policy requires the administrator of SGEEE to modify manually the automated policy(ies) to prioritize vital jobs. It is to be employed only in the most exceptional circumstances.

The first three policies are managed through the concept of tickets which, like shares, might be assigned to projects, departments, and/or users. The last policy is managed manually by the administrator.

It was agreed that the share-tree policy is to be adopted for the WRG Node 1 resource allocations.

### 4.2.4 Submitting batch jobs to SGEEE

There are two ways that you can submit jobs to this batch system:

- using **`qsub`** (command line interface) or
- using **`qmon`** (a GUI interface)

### 4.2.5 Submitting jobs using qsub

The general command to submit a job with the `qsub` is as follows:

```
% qsub [options] [script_file_name| --[script_args]]
```

To submit a job to SGEEE, you will first need to create a shell script file containing the commands to be executed by a batch request. This script must then be sent to SGEEE with the `qsub` command.

The commonly used options are:

| Option | Description |
|---|---|
| `-l h_rt=hh:mm:ss` | The wall-clock run time. |
| `-P` *`project_name`*<br><br>where `project_name` is:<br>`WhiteRose`<br>`ISS`<br>`SPEME`<br>`MechEng`<br>`Environment`<br>`Physics`<br>`Computing`<br>`Maths`<br>`FoodScience` | Specifies the project to which this job is assigned. If you do not specify this parameter your job will run under and be accounted to your default project. |
| `-help` | Prints the listing of all options |

| | |
|---|---|
| `-l h_vmem=memory` | Sets the limit on virtual memory required, for parallel jobs this limit is per processor. |
| `-pe` *`parallel_environment`* `np`<br><br>`e.g. -pe mpi_pe 8`<br>`-pe openmp 4` | This flag specifies the parallel environment i.e. use `mpi_pe` for MPI programs and `openmp` for OpenMP codes or executables built using the auto-paralleliser. The `np` parameter must be set to the number of processors. |
| `-t stop-start:stride`<br><br>`e.g. -t 2-100:2` | For array jobs, submit jobs with parameters from `start` to `stop`, incrementing by `stride`.<br>Example will submit 50 batch requests with index: 2, 4 … 98, 100. |
| `-V` | Make the environment variables from the launching shell available to the batch process. |
| `-cwd` | Execute the job from the current working directory; output files are sent to the directory from which the job was submitted, not to the home directory. |
| `-m be` | Send mail at the beginning and the end of the job to the owner |

Table 6

Descriptions of other options are available from the **`man`** pages. Options can either be specified on the command line or stored in the job submission script using lines that begin with the prefix: **`#$`**

Note that under SGEEE users should invoke the `mprun` command using the `-x` flag instead of the `-np <slots>` option. This specifies that the program should be launched using the HPC ClusterTools/Sun Grid Engine integration and automatically launches the correct number of parallel processes.

When launching OpenMP codes using the `openmp` parallel environment there is no automatic selection of the number of launched parallel threads to processors requested in the parallel environment. To configure this correctly, include the following line in the job submission script before the program is launched:

```
setenv OMP_NUM_THREADS ${NSLOTS}
```

#### *4.2.5.1 An example of a serial job submission*

For example, assuming that you had created a script file called `myjob` containing the commands you want to be executed by SGEEE, and you would like your job to be executed from your current sub-directory (`$HOME/test`) and to use not more than 3 CPU hours, 1MB of memory, and 1 processor, you may submit it with the following command:

```
% qsub -l h_rt=3:00:00 -l h_vmem=1M -cwd myjob
```

The job script file `myjob` may contain the following commands:

```
#!/bin/csh
a.out
date
```

This job will be charged to the user's default project.

During batch request submission, the script file is spooled so that subsequent changes to the original file do not affect the queued batch request. When your batch request has been submitted successfully to SGEEE, you will receive a message in the format of:

```
Your job 401  ("myjob") has been submitted.
```

Please note that you will not be able to submit your job to a named queue because the queue is determined by the resources requested by you and all jobs are waiting in a spooling area before the scheduler dispatches them to the queue. If you omit the number of processors you require, your job will be executed on a single processor.

It is also possible to set a limit on resources as part of your input script by including the options that would be given to `qsub` in lines that begin with the sequence **`#$`**.

### 4.2.6 Job output

For each batch job, two files, of the form `jobname.e`*`xxx`* and `jobname.o`*`xxx`* will be produced; the file `jobname.o`*`xxx`* contains any output that would normally be printed to the screen, and `jobname.e`*`xxx`* contains error messages.

If the `-cwd` option is specified then output files are sent to the directory from which the job was submitted, otherwise they will be dispatched to the user's home directory.

To receive email after your batch request has ended execution, please specify the `-m e` option on the `qsub` command.

### 4.2.7 An example of an MPI job submission to SGEEE

For example, assuming that you had created a script file called `myjob` containing the commands you want to be executed to run your MPI job in the C shell by SGEEE, and you would like your job to be executed from your current directory and be informed via email of the start and the end time of your job, and to use not more than 10 CPU minutes per processor, and 4 processors, you may submit it with the following command:

```
% qsub myjob
```

The job script file `myjob` may contain the following commands:

```
#$ -P ISS
#$ -cwd
#$ -m be
#$ -l h_rt=:10:
#$ -pe mpi_pe 4
cd $HOME/test
mprun -x sge mympijob
date
exit
```

Please note that the `-x sge` flag must be given to `mprun`. The main purpose of this is so that Sun Grid Engine is able to control the batch job. This flag also controls which nodes will run the MPI job which, for efficiency, will limit all MPI processes to be launched on the same shared-memory node. Failure to include this flag will result in different maxima nodes running different MPI processes and a significant drop in performance will be incurred.

The effect of running the above batch script would be to change directory to the test subdirectory in your home directory; then the user's program (`mympijob`) would be executed on 4 processors and the date added at the end of your output file.

### 4.2.8 An example of OpenMP job submission to SGEEE

For example, you want to run an OpenMP program spawning 8 processes and you would like your output to be returned to your current directory from which you submit your job, and to use not more than 20 CPU minutes and 1GB of memory per processor. Assuming that you had created a script file called `myjob` containing the commands you want to be executed by SGEEE, you may submit it with the following command:

```
% qsub -l h_rt=:20:00 -l h_vmem=1G -cwd -pe openmp 8 myjob
```

The job script file `myjob` may contain the following commands:

```
#!/bin/csh
cd $HOME/test
setenv OMP_NUM_THREADS ${NSLOTS}
myopenmpjob
date
exit
```

Please note that the number of parallel threads to be launched must be included in the batch submission script through the line "`setenv OMP_NUM_THREADS ${NSLOTS}`" as there is no automatic selection of parallel threads to requested runslots (CPUs). Failure to include this line will result in the job running on a single processor.

### 4.2.9 An example of array job submission to SGEEE

SGEEE contains a convenient facility for launching a set of jobs that consist of parameterized and repeated execution of the same set of operations. A simple example of this is when undertaking several program runs each of which uses slightly different command line or input parameters. This array job facility therefore provides a convenient notation for submitting a single job submission script that runs many batch requests.

To specify the list of jobs that should be run the "`-t start-stop:stride`" flag should be passed to `qsub`, where `start`, `stop` and `stride` indicate the initial, final and incremental values of the jobs to be run respectfully. Within the job submission script the `${SGE_TASK_ID}` variable can be used to determine the index number of the particular array job.

Array jobs can be both parallel and serial, e.g. an example array MPI job script could consist of the following:

```
# Use current working directory
#$ -cwd
# run MPI job on 4 processors
#$ -pe mpi_pe 4
# request 1 hour runtime
#$ -l h_rt=1:00:00
# input file is data.<index>
# run program
mprun -x sge mpi_prog data.${SGE_TASK_ID}
```

Thus to run the program using the files `data.1`, `data.2` … `data.100` the following command should be used to submit the array job to the batch queues:

```
% qsub -t 1-100:1 array.csh
```

## 4.3 Interactive SGEEE jobs

SGEEE is configured to allow both parallel and serial jobs to be launched interactively using the queues. Interactive jobs will normally be run right away, or not at all – if the necessary resources are not currently available; however the `-now [y|n]` flag may be used to override this. There are three commands that enable this which take many of the options available to `qsub`:

| | |
|---|---|
| `qrlogin` | Queued telnet session (limited use). |
| `qsh` | Queued interactive X-windows shell. |
| `qrsh` | Queued interactive execute command. |

e.g. to launch a serial, interactive X-windows shell for 4 hours:

```
% qsh -cwd -l h_rt=4:00:00
```

For MPI jobs the SGE integration can be used to launch the program; i.e. to launch a 4 processor, interactive MPI job for 4 hours runtime specify:

```
% qrsh -pe mpi_pe 4 -cwd -l h_rt=4:00:00 mprun -x sge ./mpiprog
```

For OpenMP interactive jobs please note that there is no automatic specification of the number of parallel threads that will be executed unless the queued shell knows about the variable `$OMP_NUM_THREADS`. In a queued interactive X-windows shell (`qsh`) this can be done by typing:

```
% setenv OMP_NUM_THREADS ${NSLOTS}
```

into the terminal console window that appears. Alternatively if `$OMP_NUM_THREADS` is already set in your current shell, you can export all variables to the queued shell by specifying the `-V` option i.e.:

```
% setenv OMP_NUM_THREADS 4
% qrsh -V -cwd -l h_rt=4:: -pe openmp ${OMP_NUM_THREADS} ./ompprog
```

will launch the interactive job `ompprog` on 4 processors, for 4 hours runtime.

It sometimes may be necessary to use the `-V` flag to export variables such as your display, or license variables for running particular applications for example, to run the application Fluent[†] via the interactive queues, provided you have set the necessary licensing environment, the following line will enable the Fluent GUI to launch and run a 3d case for 4 hours on 4 processors:

```
% qrsh -pe mpi_pe 4 -cwd -l h_rt=4:00:00 -V fluent -sge 3d
```

Further information on launching application software via the batch queues can be obtained from: http://www.leeds.ac.uk/iss/wrgrid/Documentation/Node1software.html.

## 4.4 Querying queues

The `qstat` command may be used to display information on the current status of Grid Engine jobs and queues. The basic format for this command is:

```
% qstat [switches]
```

Important switches are as follows:

| Switch | Action |
|---|---|
| `-help` | Prints a listing of all options. |
| `-f` | Prints a summary on all queues. |
| `-U` *username* | Displays status information with respect to queues to which the specified users have access. |

Table 7

The switches are documented in the **`man pages;`** for example, to check all options for the `qstat` command type:

```
%  man qstat
```

[†] Please note that the launching of commercial applications such as Fluent is only available to users that have access to either a private or departmental license for the application. Fluent is a registered trademark of Fluent Inc.

## 4.5 Job deletion

To delete your job issue the following command:

```
% qdel jobid
```

where `jobid` is the numerical tag that is returned by SGEEE when the job is submitted. This is also available from the `qstat` command. To force action for running jobs issue the following command:

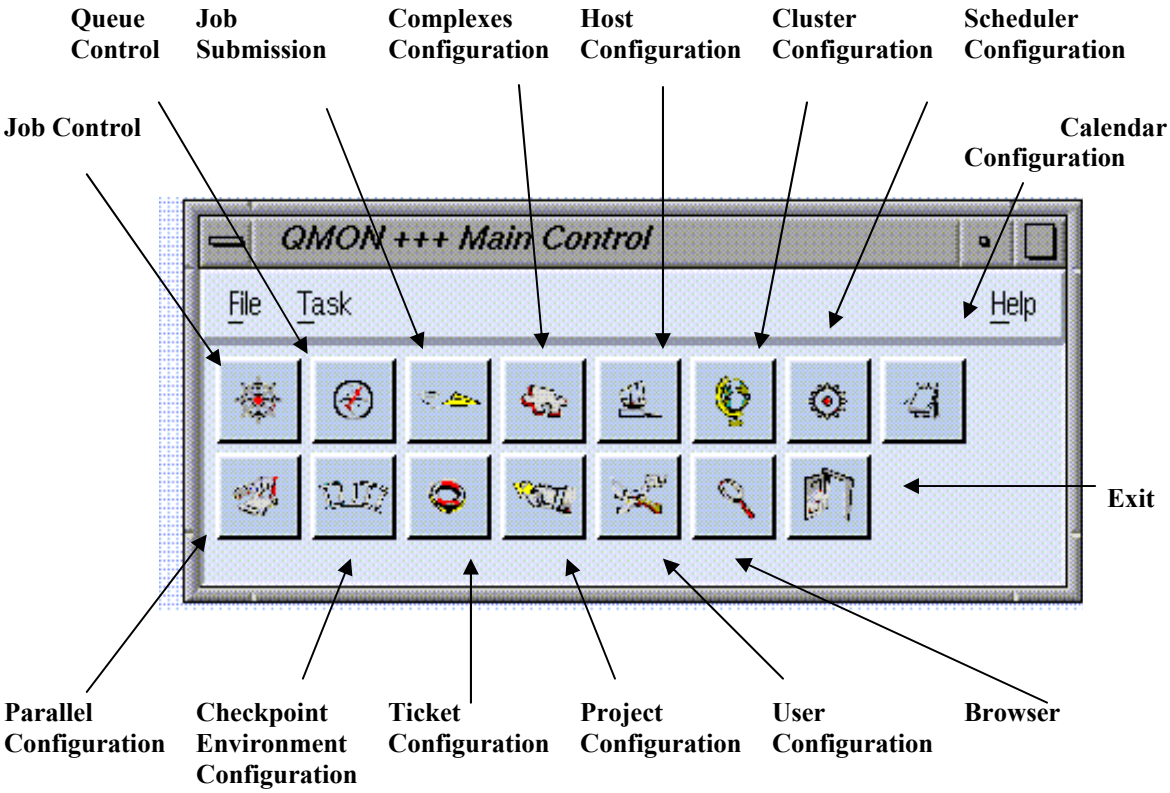
```
% qdel -f jobid
```

## 4.6 The GUI `qmon` command

You can run Grid Engine using a Graphical User Interface (GUI). Perhaps this is a simple way of using SGEEE. The GUI, which runs in X-Windows, is called by typing the command:

```
%  qmon
```

The main control window will offer you a number of buttons to submit your job, to query its status, to suspend its execution or to remove it altogether.




## 4.7 Usage accounting statistics

The SGEEE system is set up to generate accounting statistics for jobs run under this product. This information is reported on a per-month per-department basis at http://www.leeds.ac.uk/iss/wrgrid/Usage.

# 5 On-line Information

There are various forms of on-line information available, including documentation at: http://www.leeds.ac.uk/iss/wrgrid/Documentation. Unix on-line man pages may be accessed by typing:

```
% man topic
```

Sun technical manuals are available on the Web at `http://docs.sun.com`. For further information on SGEEE please see the following URL:
`http://gridengine.sunsource.net/`

# 6 Help and user support

General user queries and further guidance on the use of the system may be obtained via email to `helpdesk@leeds.ac.uk`. This is the preferred way of dealing with users' queries. However, users who require direct user-support may arrange it by sending email to helpdesk@leeds.ac.uk

# 7 Emailing list

All new users are subscribed to an emailing list for users of the facility. This is a moderated list and users are encouraged to use it as a discussion forum for problems common to high performance computing and grid technology areas.

To disseminate information to this list please send email to:
`WRG-maxima-users@leeds.ac.uk`

# 8 Code of Conduct

The Code of Conduct is a set of rules of etiquette, which the Consortium agreed to adopt; in order to make the most effective use of the system, and to ensure that resources are fairly shared between shareholders. The following rules were agreed:

- Users are reminded that an SGEEE shell script file should not contain a loop spawning further jobs.
- Users are asked to run interactive jobs via SGEEE's queues and not directly. This is to ensure adequate response for batch and interactive jobs.
- Users are reminded that they should not put huge amounts of data to standard output. This may cause the spooling system to fill up and make other jobs fail.
- Users are asked to use their own checkpointing facility so their work is not lost in the event of an unexpected system crash.
- Users are reminded that the `/tmp` and `/var/tmp` directories may be purged at any time and they should not be used to store data.

# 9 Hints

- All users are strongly advised to checkpoint their long running programs.
- The simplest way to gather basic data about program performance and resource utilisation is to use the `time(1)` command or in `csh` the `set time` command

- To display the top processes running type `/usr/ucb/ps -aux | head`
- Solaris `prstat` command is equivalent to the `top` command in IRIX.
- Use the **`prstat -U`** `username` command to display active process statistics for the specified username.
- To use the NAG library specify the flag `-lnag` on the compilation line.

# 10 Links*

The following links might be useful:

| | |
|---|---|
| A Listing of Parallel Computing Sites maintained by David A.Bader | `http://computer.org/parascope/` |
| Edinburgh Parallel Computing Centre | `http://www.epcc.ed.ac.uk/` |
| CSAR high performance service at Manchester Computing | `http://www.csar.man.ac.uk/` |
| Message Passing Interface Forum | `http://www.mpi-forum.org/` |
| OpenMP Forum | `http://www.openmp.org/` |
| The Sun documentation Web site | `http://docs.sun.com` |
| The Sun documentation for Forte Developer products | `http://www.sun.com/forte/developer` |
| The Sun documentation for HPC Cluster Tools | `http://www.sun.com/hpc` |
| The Sun White Paper: "Delivering Performance on Sun: Optimising Applications for Solaris Operating Environment" | `http://www.sun.com/software/whitepapers.html` |
| The Sun GridEngine: | `http://wwws.sun.com/software/gridware/` |

*/ Please note that these links are provided for convenience only; the author of this page or ISS do not necessarily endorse the views or products mentioned in them.

# Appendix A

This Appendix contains a summary of some useful Unix commands.

| | |
|---|---|
| **apropos** | Displays the main page name section number, and a short description for each man page whose NAME line contains keyword |
| **cat** | Reads each file in sequence and writes it on the standard output |
| **cd** | Changes working directory |
| **cmp** | Compares two files |
| **cp** | Copies the contents of `source_file` to the destination path named by `target_file` |
| **date** | Gets the current date and puts it into the character string str. The form is `dd-mm-yy` |
| **diff** | Compares the contents of `file1` and `file2` and write to standard output a list of changes necessary to convert `file1` into `file2` |
| **exit** | Terminates process with status. |
| **finger** | Displays in multi-column format information about each logged-in user, e.g. user name and login time |
| **grep** | Searches text files for a pattern and prints all lines that contain that pattern |
| **history** | View list of the last commands |
| **jobs** | Shows your background jobs |
| **kill** | Sends a terminate signal to a process, not necessarily killing the process |
| **ls** | Lists the contents of the directory |
| **man** | Displays information from the reference manuals |
| **mkdir** | Creates a new directory |
| **more** | Displays the contents of a text file on the terminal, one screenful at a time |
| **mv** | Renames (or moves) a file |
| **nohup** | The background process is continued after logout |
| **passwd** | Changes the password |
| **pg** | Display your file, one screenful at a time |
| **pwd** | Returns the current working directory name |
| **qsub** | Submits batch jobs to the Grid Engine queuing system |
| **qstat** | Shows the current status of the available Grid Engine queues and the jobs associated with the queues |
| **qdel** | Provides a means for a user/operator/manager to delete one or more jobs |

| | |
|---|---|
| **rm** | Removes (delete) files |
| **rmdir** | Removes (delete) directory |
| **spell** | Checks a file for spelling mistakes |
| **tar** | Archives and extracts files to and from a single file called a `tar` file |
| **write** | Reads lines from the user's standard input and writes them to the terminal of another user |